DATA TYPES AND OPERATIONS

DATA TYPES

Constant

A constant is **a fixed value** of a data type that **cannot be changed**

- Integer Constants

  Whole numbers → Do not have decimal points

  Examples: 83 9 25 178 -13 0

- Real Constants

  Numbers that have decimal points

  Examples: 2.3 -5.6 0.42E9 0.58E-6 3. 0.

Constants

- Logical Constants

  Two values

  .TRUE.

  .FALSE.

- Character Constants

  One character or string of characters between two single quotes

  'THIS IS CHAPTER TWO‘

  'ISN''T IT?'

Variables

- **Occupies a place** in the computer’s memory
- Must have a **name** to be referenced later
- Its **value could be changed**
- May be of different types
  - Integer
  - Real
  - Logical
  - Character

Variable Names

There are some rules for choosing variable names in FORTRAN

- Should **start** with an alphabetic character (A, B, C,... ,Z)
- Its **length** should **not exceed 6** characters
- Could contain digits (0, 1, 2,...., 9) but not the first character
- Should not contain special characters
- Should not contain blanks

Variables

- Integer Variables

  Can hold only integer values

  Can be defined using **INTEGER** statement

  Examples:

  ```
  INTEGER  A, B, X, NUM
  ```

- Real Variables

  Can hold only real values

  Can be defined using **REAL** statement

  Examples:

  ```
  REAL  X, Y, Z
  ```

Variables

- Implicit definition
  - it is a good practice to explicitly define all variables used in your program
  - Variables that are assigned values but **not defined** will be assumed to be of **REAL** type unless the variable name starts with any of the following letters:

    I J K L M N

  - if the variable name starts with

    I J K L M N

    and not defined , it will be assumed as **INTEGER**

Variables

- Logical Variables

  Can only have logical values

  Values can be

  - .TRUE.
  - .FALSE.

  Can be defined using **LOGICAL** statement

  Example:

```
LOGICAL  FLAG, TEST, FLAG1
```

Variables

- **Character Variables**

Can hold only character values

Can be defined using **CHARACTER** statement

The length can be defined , otherwise will be assumed as 1

Examples:

```
CHARACTER   NAME*10
CHARACTER   T1 , T2
CHARACTER   A*8 , B
CHARACTER*5  Z , Z1 , Z2
CHARACTER*7  Z , Z1*3 , Z2
```

Arithmetic Operations

Addition	Subtraction	Multiplication	Division	Exponentiation
+	-	*	/	**

Operators: +   -   *   /   **

Examples:

X – Y

X + Y – 4 / Z

– A + B – C

Priority

(   )

**

*   /

+   -

Arithmetic Operations

- Integer Operations

  The result of arithmetic operations with both operands as integer is integer

  Examples:

  70 – 31   3**2   8 / 3

- Real Operations

  The result of arithmetic operations with both operands as real is real

  Examples:

  70.0 – 31.0   3.0**2.0   8.0 / 3.0

Arithmetic Operations

- Mixed-mode Operations

The result of an arithmetic operation with one integer operand and one real operand is real

Examples:

70.0 – 31	3**2.0	8.0 / 3
70 – 31.2	3.5**2	8 / 3.0

Examples

- Example 1: Evaluate the following arithmetic expression

```
20 - 14 / 5 * 2 ** 2 ** 3
```

- Example 2: Evaluate the following arithmetic expression

```
14.0 / 5 * (2 * (7 - 4) / 4) ** 2
```

Examples

- Example 3: Rewrite the following FORTRAN expression as a mathematical form

```
X + Y / W – Z
```

- Example 4: Rewrite the following FORTRAN expression as a mathematical form

```
X ** (1.0 / 2.0) / Y ** Z
```

- Example 5: Convert the following mathematical expression into FORTRAN expression. Use minimum number of parenthesis

$$\frac{\sqrt{a+b}}{a^2 - b^2}$$

Logical Operations

Logical Operations evaluate to either **.TRUE.** or **.FALSE.**

- Logical Operators

  .AND. .OR. .NOT.

  Example:

  .FALSE. .OR. .NOT. .TRUE. .AND. .TRUE.

- Relational Operators

  - The values of arithmetic expressions can be compared using relational operators
  - The result of a relational operation is **.TRUE.** or **.FALSE.**
  - Relational Operators:

    .EQ. .NE. .GT. .GE. .LT. .LE.

  Examples:

  X .EQ. Y Z + A .GT. X

Logical Operations

Logical Expressions evaluate to either **.TRUE.** or **.FALSE.**

Example 1: Given that X has a value of 3.0, Y has a value of 5.0, Z has a value of 10.0, and FLAG is a logical variable with .FALSE. Value, evaluate the following FORTRAN expression:

```
.NOT.FLAG .AND. X*Y .GT. Z .OR. X+Y .GT. Z
```

Priority

Arithmetic expressions

Relational expressions

Logical expressions

```
.NOT. FLAG .OR. FLAG

.NOT. FLAG . AND. FLAG

.NOT. .NOT. FLAG

 X .GT. Y – Z / 2.0
```

Assignment Statement

The Assignment Statement in FORTRAN assigns a value to a variable. The general form is:

variable = expression

Expression must have a value of the same type as the variable

Exception

- integer values can be assigned to real variables
- real values can be assigned to integer variables

Example:

```
INTEGER  M , N
REAL  A , B
A = 6.5
B = A + 9/2
M = B
N = B + 3.5
A = N
A = M + N
N = A + B
M = N + 3 **3.0
A = B + M
```

Example:

```
M = 9.5
J = M/2*2
N = J*2/3
A = N + 2.5
B = A /2*6
L = N*2/3 + B
C = L + 3.5/2.5
K = C + J /4.5*9
D = 25**(1/2)
E = 8**1/3
```

Input Statement

READ*, list of variables separated by commas

Note the followings

- **each reading** statement starts reading **from a new line**
- reading **continues** from the **next line** if the input data is **not enough**
- data values in a line should be separated by **commas** or **blanks**
- data values must agree in types with the variables they are read into
  - except that integer values **can** be read into real variables
  - but real values **can not** read into integer variables
- Extra data on an input line is ignored

Output Statement

PRINT*, list of variables , expressions, or constants separated by commas

Note the followings

- **each PRINT** statement starts printing on a **new line**
- printing continues in the next line if the line is not enough to hold the output of the print statement
- a variable that does not have a value will produce ???? if it is printed

A Complete Program

The following program reads three real numbers, prints them, computes their average and prints it:

```
C     THIS PROGRAM READS 3 REAL NUMBERS
C     AND COMPUTES AND PRINTS THE AVERAGE
C
      REAL  NUM1, NUM2, NUM3, SUM, AVG
      PRINT*, 'ENTER THREE REAL NUMBERS'
      READ*, NUM1, NUM2, NUM3
      PRINT*, 'THE NUMBERS ARE', NUM1, NUM2, NUM3
      SUM = NUM1 + NUM2 + NUM3
      AVG = SUM /3.0
      PRINT*, 'THE AVERAGE IS', AVG
      END
```